STA130H1F

Class #8 - Classification Trees

Prof. Nathalie Moon

2018-11-12

Plan for today's class

1. Supervised vs unsupervised learning
2. Classification trees
    - Interpretation
    - Methodology
    - Training and testing
    - Accuracy
3. ROC curves

Reference (available on Quercus) Baumer, Benjamin S., Daniel T. Kaplan, and Nicholas J. Horton. Modern data science with R. CRC Press, 2017. Pages 173-180, 189-192.

Learning from Data

"Machine learning" / "Deep learning"

- Objective: discover patterns in data
- How might this be done?

Supervised and unsupervised learning

- Supervised learning:
  - Data is labelled
  - Algorithm is "learning" from the labelled data, like a teacher "supervises" students' learning
  - We know the true answer; the algorithm stops when it achieves an acceptable level of performance

Supervised and unsupervised learning

- Supervised learning:
  - Data is labelled
  - Algorithm is "learning" from the labelled data, like a teacher "supervises" students' learning
  - We know the true answer; the algorithm stops when it achieves an acceptable level of performance
- Unsupervised learning:
  - Goal is to discover the underlying structure/distribution in the data
  - We don't know the "true answer": there is no teacher to "supervise" the process
  - Example: clustering analysis

Supervised learning

In this class: we'll only focus on supervised learning

- Response variable: the thing we want to predict (y)
  - also known as: output, label, or dependent variable
- Predictor(s) $x_1, x_2, \ldots, x_p$: the variable(s) we want to base the prediction on
  - also known as: features, covariates, inputs, independent variables

Use xs to predict y.

Types of supervised learning

Categorical response:

ex: men/women, nationality, ...

⇒ classification trees

Continuous response:

ex: height, weight, grades, ...

⇒ linear regression.

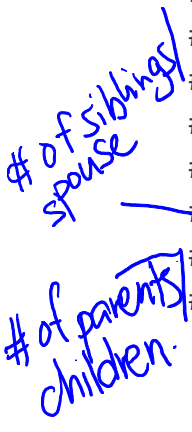
Example: Titanic

Sample of passengers of the Titanic

```
titanic <- "https://goo.gl/At238b" %>%
  read_csv %>% # read in the data
  select(survived, embarked, sex,
         sibsp, parch, fare) %>%
  mutate(embarked = factor(embarked),
         sex = factor(sex)) %>%
  mutate(survived = ifelse(survived==1, "yes", "no"))

glimpse(titanic)
```

```
## Observations: 1,309
## Variables: 6
## $ survived <chr> "yes", "yes", "no", "no", "no", "yes", "yes", "no", "...
## $ embarked <fct> S, S, S, S, S, S, S, S, S, C, C, C, C, S, S, S, C, C,...
## $ sex      <fct> female, male, female, male, female, male, female, mal...
## $ sibsp    <int> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0,...
## $ parch    <int> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,...
## $ fare     <dbl> 211.3375, 151.5500, 151.5500, 151.5500, 151.5500, 26....
```



Example: Titanic

Can we use the variables in this dataset to predict which passengers survived / didn't survive?

Which variables would you like to try?

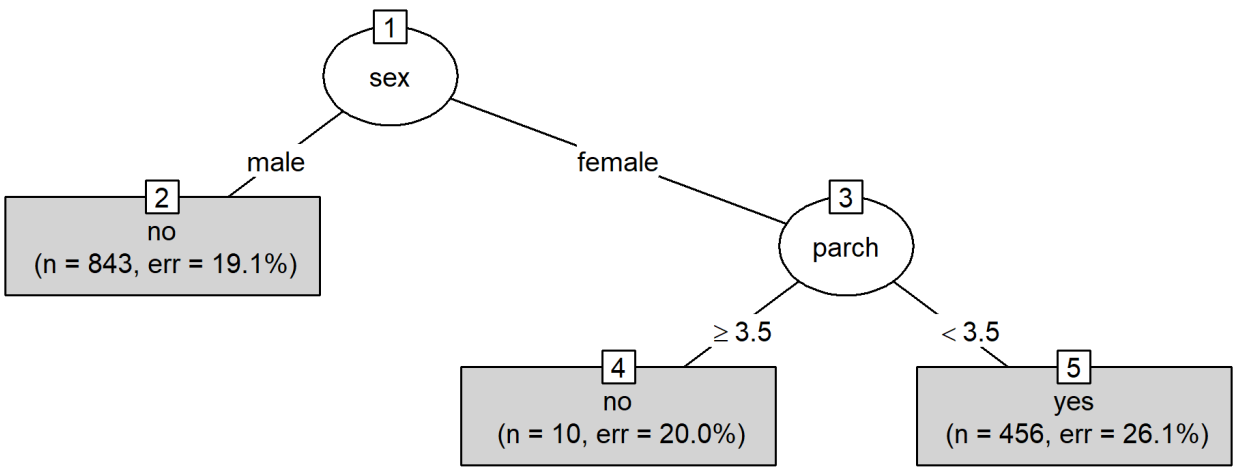
Example: Titanic

Can we use the variables in this dataset to predict which passengers survived / didn't survive?

Which variables would you like to try?

```
library(rpart)
library(partykit)

rtree_fit <- rpart(survived ~ sex + parch, titanic)
plot(as.party(rtree_fit), type="simple", gp=gpar(cex=0.8))
```



Interpreting a classification tree

- Nodes: variable + decision rule

Interpreting a classification tree

- Nodes: variable + decision rule
- Terminal node: prediction
  - what is n?
  - what is err?

"decision" nodes are oval → rectangular

⇒ # of obs. that end up in that bucket when you answer the questions in the tree

↳ error rate: % of incorrect predictions.

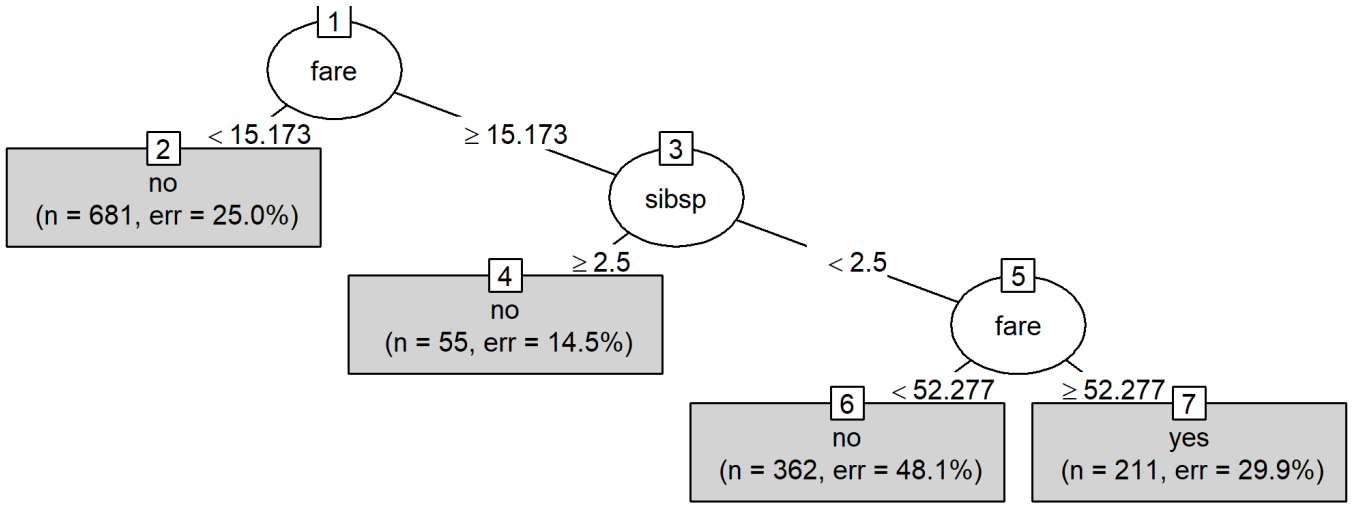
Interpreting a classification tree

- Nodes: variable + decision rule
- Terminal node: prediction
  - what is n?
  - what is err?
- How to use the tree to make a prediction for a new observation?

↳ start at the top.
Ask the first question and follow the appropriate branch of the tree
Repeat until you reach a terminal node
↳ get the prediction

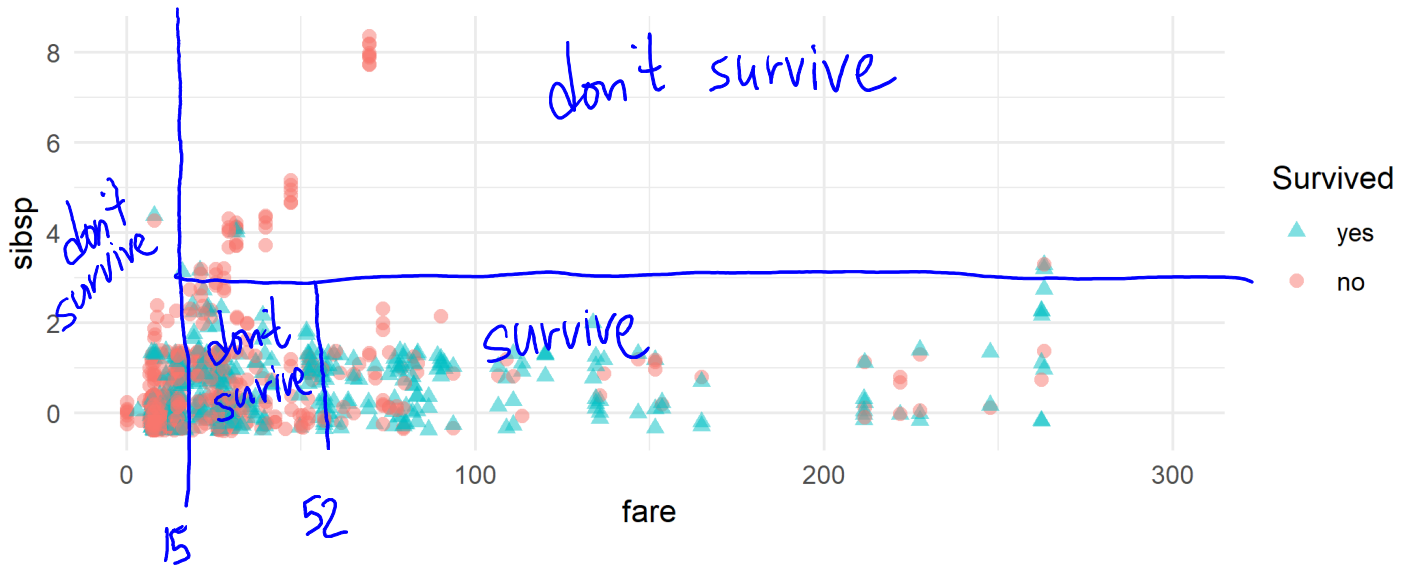
Geometric interpretation of a classification tree

```
rtree_fit <- rpart(survived ~ fare + sibsp, titanic,
                   control = rpart.control(cp=0.02))
plot(as.party(rtree_fit), type="simple", gp=gpar(cex=0.8))
```



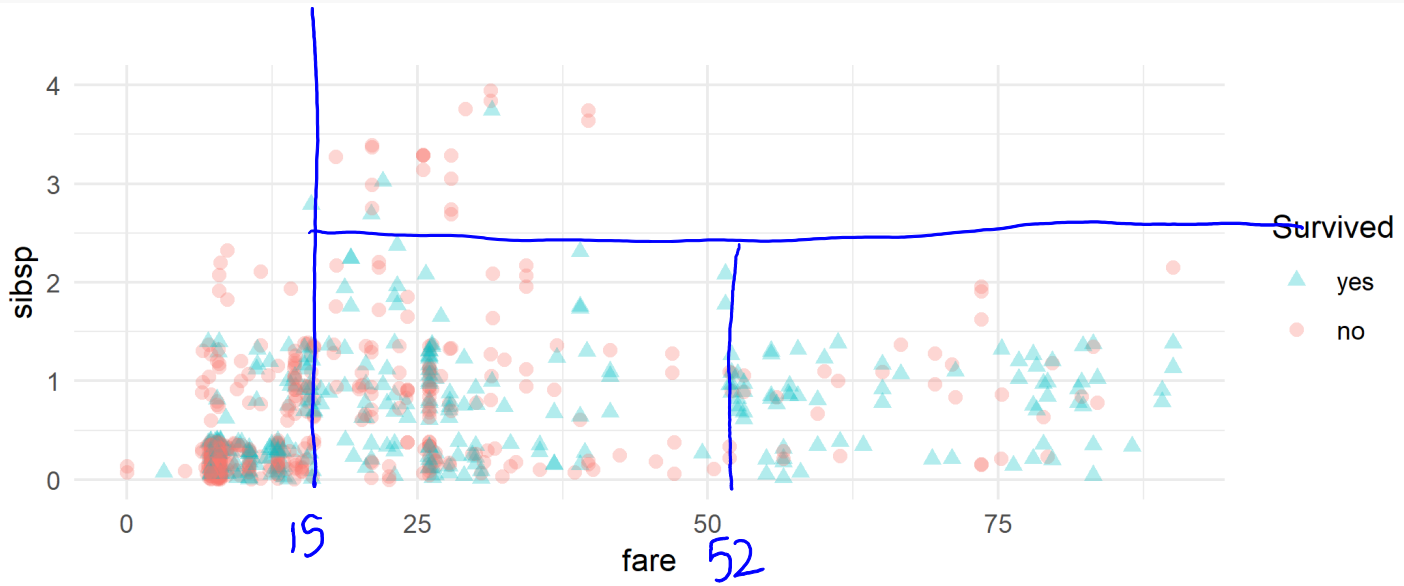
Geometric interpretation of a classification tree

```
ggplot(titanic,
       aes(fare, sibsp, shape = factor(survived),
           colour = factor(survived))) +
 geom_point(size = 2, position="jitter", alpha=0.5) + xlim(0,300) +
 theme_minimal() +
 scale_color_discrete(name = "Survived", breaks = c("yes", "no")) +
 scale_shape_discrete(name = "Survived", breaks = c("yes", "no"))
```



Geometric interpretation of a classification tree: Zooming in

```
ggplot(titanic,
       aes(fare, sibsp, shape = factor(survived),
           colour = factor(survived))) + ylim(0,4) +
 geom_point(size = 2, position="jitter", alpha=0.3) + xlim(0,90) +
 theme_minimal() +
 scale_color_discrete(name = "Survived", breaks = c("yes", "no")) +
 scale_shape_discrete(name = "Survived", breaks = c("yes", "no"))
```



What do we need to build a classification tree?

What do we need to build a classification tree?

- a response variable (y) ⇒ the thing we want to predict
- A set of candidate predictors ⇒ your x variables
- A method to evaluate if a split is "good"
- A rule to stop splitting
- A rule for assigning each terminal node to a category

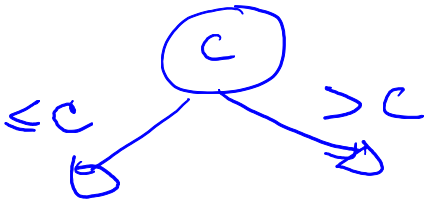
What kind of questions can we ask?

Numerical predictors:

pick a threshold (c).

↳ bigger than c

↳ less than c



Categorical predictors:

↳ define sets of values : {NA, SA, Europe}

{Asia, Africa}

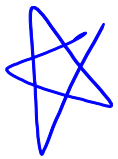
All questions need to binary. (two choices)

↳ yes/no

What kind of questions can we ask?

Numerical predictors:

Categorical predictors:



Key feature: all questions must be binary (yes/no)

Method to build a classification tree

At each node, the algorithm searches through the predictors one at a time

1. For each variable: find the "best" split
2. Compare the best split for each variable
3. Select the best of the best as the decision rule
4. Repeat steps 1-3 for each of the child nodes

Principles for building a classification tree

Two goals:

1. We want each node to be as "pure" as possible
2. Don't want to have a tree that is too complex

↳ less useful in terms of interpretation.

↳ risk of overfitting ⇒ too specific to the data used to fit the tree.

Impurity

A node is **impure** if all classes are "equally" mixed together (or at least, very mixed together)

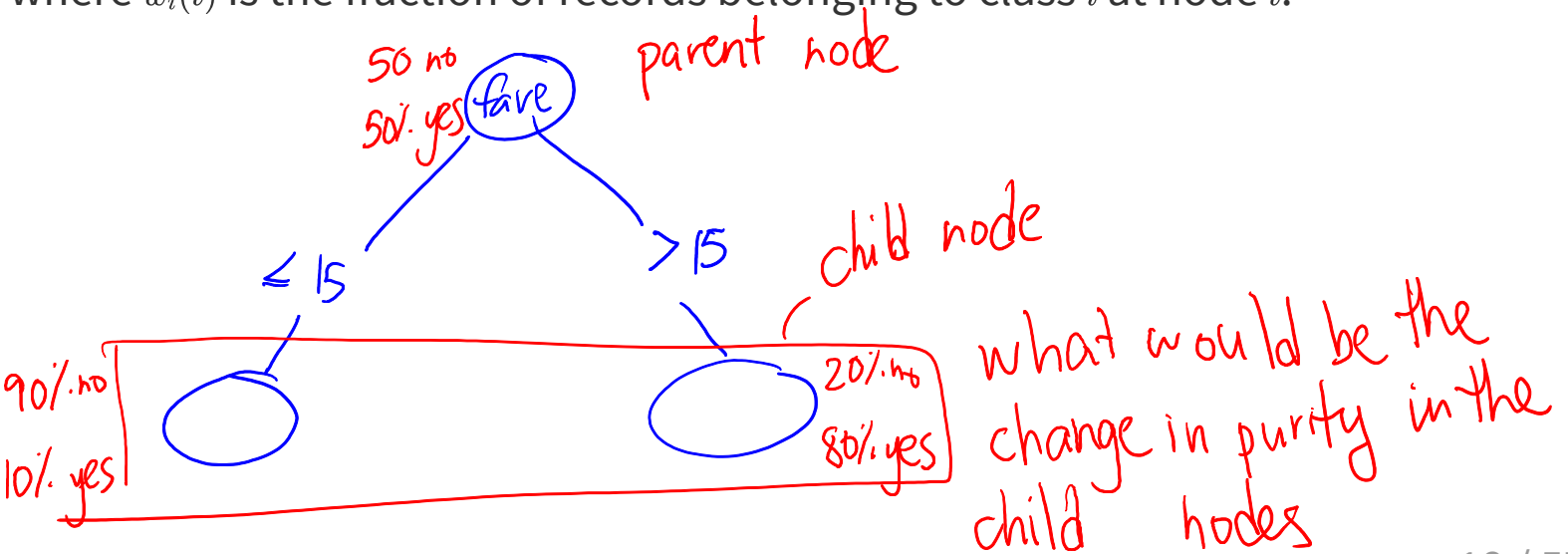
There are several measures of impurity

$$Gini(t) = 1 - (w_1(t))^2 + (w_2(t))^2$$

→ default used by rpart(.)

$$Entropy(t) = -w_1(t)\log_2(w_1(t)) + w_2(t)\log_2(w_2(t))$$

where $w_i(t)$ is the fraction of records belonging to class i at node t.



Splitting and stop-splitting rules

A split is good if it leads to a large decrease in impurity, ΔI.

Splitting and stop-splitting rules

A split is good if it leads to a large decrease in impurity, ΔI.

The maximum decrease in impurity is the "best" split.

Splitting and stop-splitting rules

A split is good if it leads to a large decrease in impurity, ΔI.

The maximum decrease in impurity is the "best" split.

A simple stop-splitting rule is to set a threshold (e.g. $\beta > 0$) and say a node is terminal if $\Delta I < \beta$

↳ to weigh the value of increasing purity with the added complexity of the tree/overfitting

Example: Predicting high income earners

```
census <- read.csv("http://archive.ics.uci.edu/ml/
              machine-learning-databases/adult/adult.data",
              header = FALSE)
names(census) <- c("age", "workclass", "fnlwgt",
                   "education", "education.num",
                   "marital.status", "occupation",
                   "relationship", "race", "sex",
                   "capital.gain", "capital.loss",
                   "hours.per.week", "native.country",
                   "income")
```

Example: Predicting high income earners

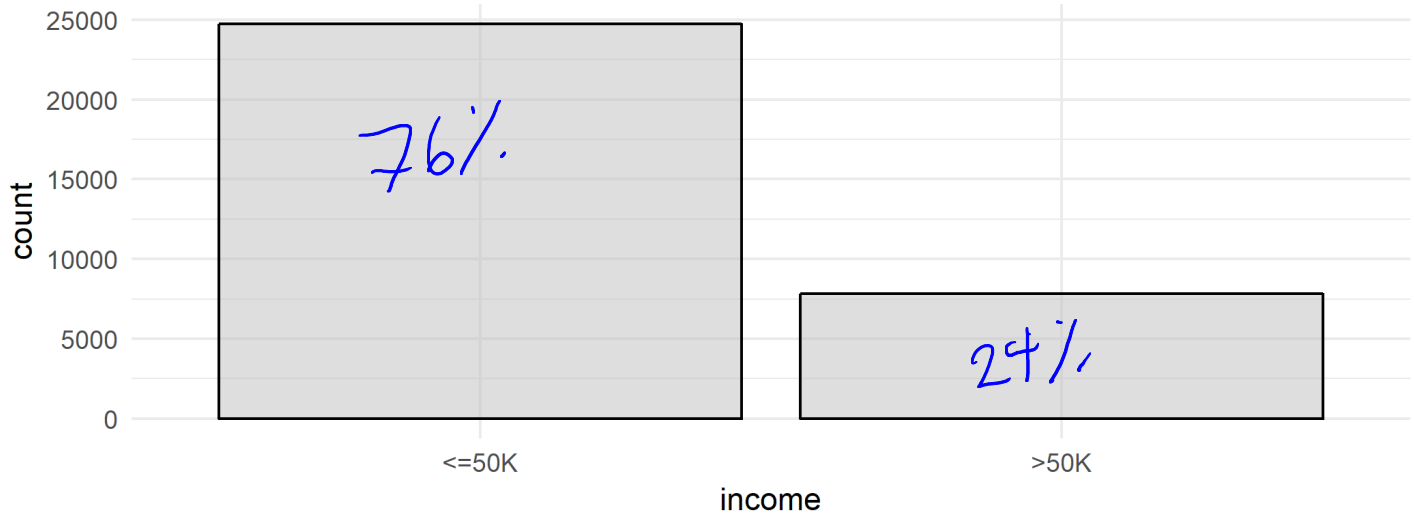
```
glimpse(census)
```

```
## Observations: 32,561
## Variables: 15
## $ age            <int> 39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 37, 30,...
## $ workclass      <chr> "State-gov", "Self-emp-not-inc", "Private", "Pr...
## $ fnlwgt         <int> 77516, 83311, 215646, 234721, 338409, 284582, 1...
## $ education      <chr> "Bachelors", "Bachelors", "HS-grad", "11th", "B...
## $ education.num  <int> 13, 13, 9, 7, 13, 14, 5, 9, 14, 13, 10, 13, 13,...
## $ marital.status <chr> "Never-married", "Married-civ-spouse", "Divorce...
## $ occupation     <chr> "Adm-clerical", "Exec-managerial", "Handlers-cl...
## $ relationship   <chr> "Not-in-family", "Husband", "Not-in-family", "H...
## $ race           <chr> "White", "White", "White", "Black", "Black", "W...
## $ sex            <chr> "Male", "Male", "Male", "Male", "Female", "Fema...
## $ capital.gain   <int> 2174, 0, 0, 0, 0, 0, 0, 0, 14084, 5178, 0, 0, 0...
## $ capital.loss   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ hours.per.week <int> 40, 13, 40, 40, 40, 40, 16, 45, 50, 40, 80, 40,...
## $ native.country <chr> "United-States", "United-States", "United-State...
## $ income         <chr> "<=50K", "<=50K", "<=50K", "<=50K", "<=50K", "<...
```

We want to predict which individuals are high income earners

Example: Predicting high income earners

```
census %>% ggplot(aes(income)) +
  geom_bar(colour = "black", fill = "grey", alpha = 0.5) +
  theme_minimal()
```



What trivial prediction could we make? (sometimes called null model)

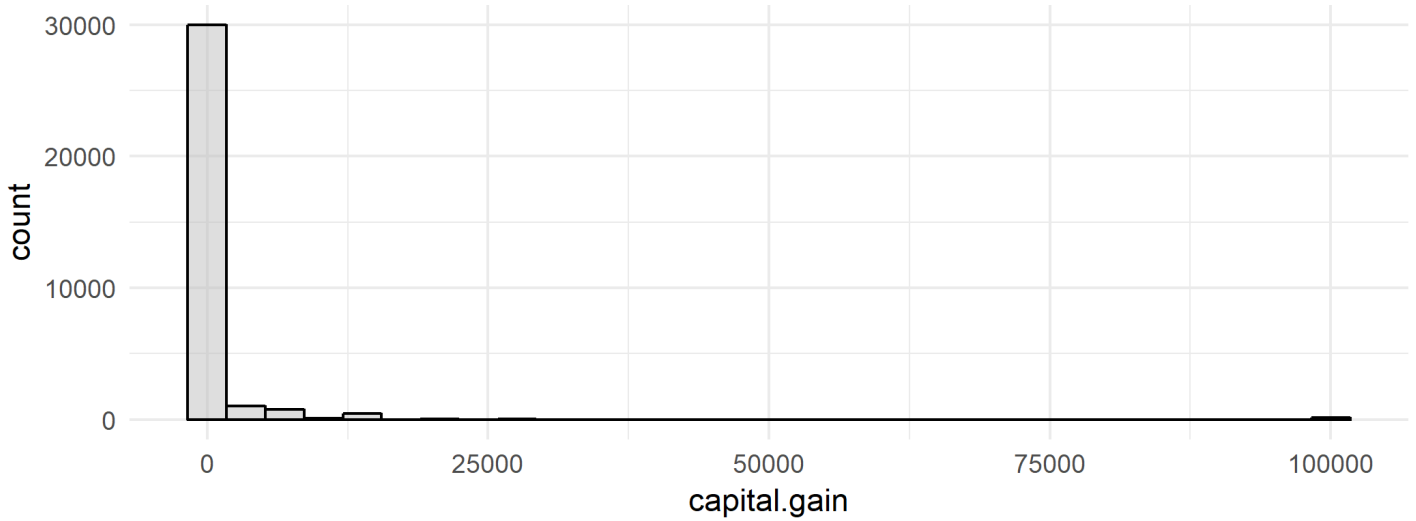
↳ always predict <50K

What would the accuracy be?

↳ we'd be correct 76% of the time!

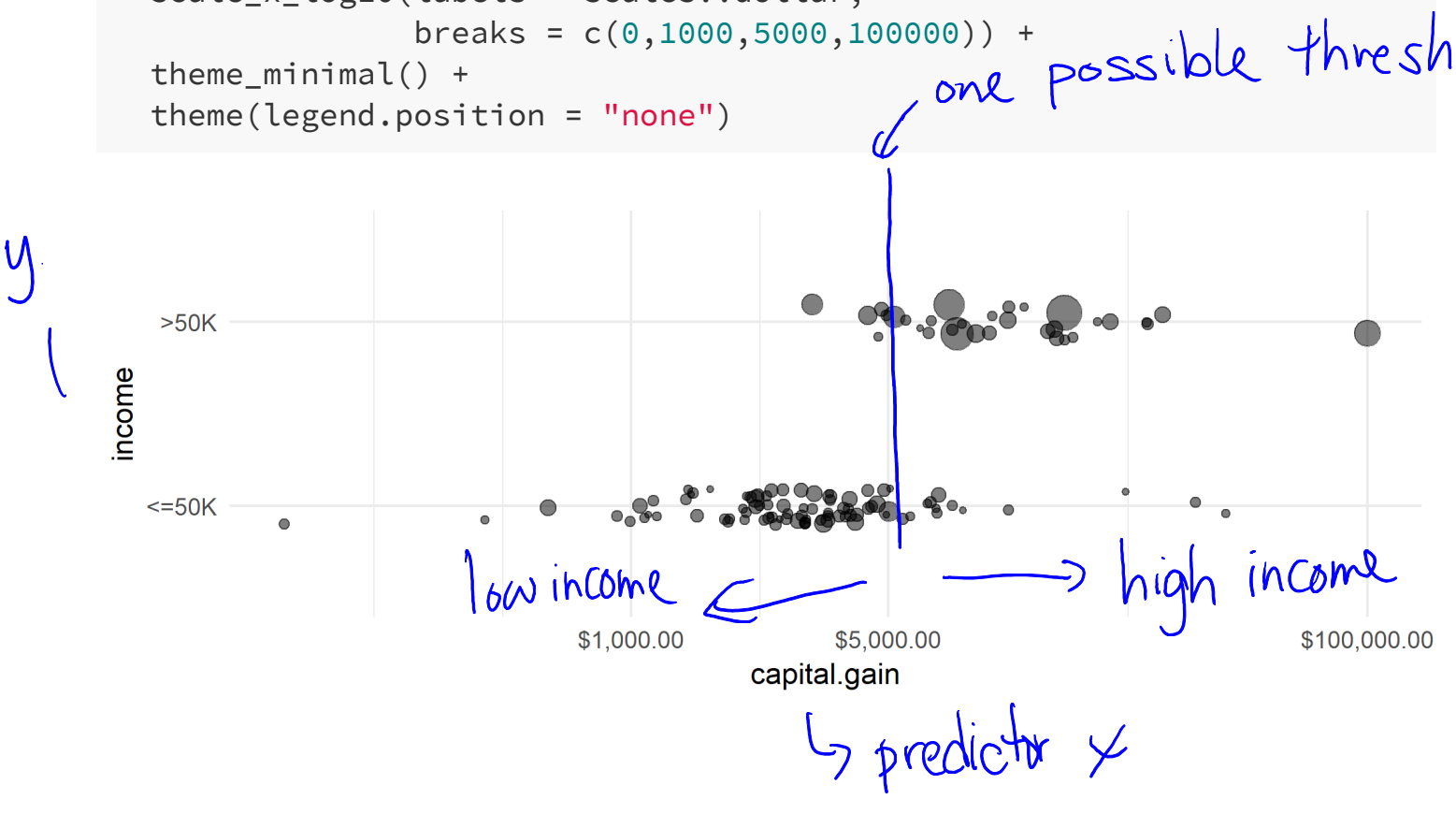
Example: Predicting high income earners

```
census %>% ggplot(aes(capital.gain)) +
  geom_histogram(colour = "black", fill = "grey", alpha = 0.5) +
  theme_minimal()
```



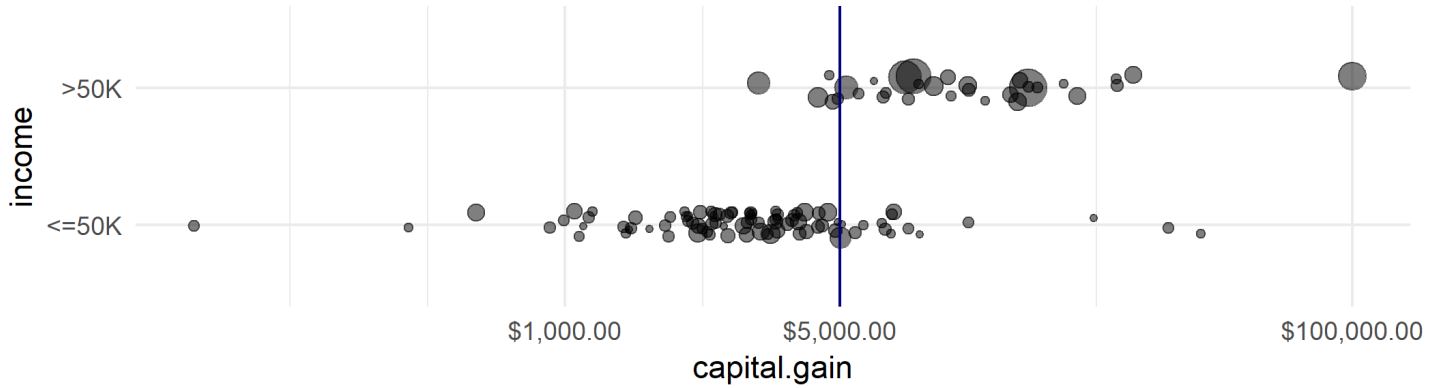
Example: Predicting high income earners

```
census %>% ggplot(aes(x = capital.gain, y = income)) +
  geom_count(position = position_jitter(width = 0, height = 0.1),
             alpha = 0.5) +
  scale_x_log10(labels = scales::dollar,
                breaks = c(0,1000,5000,100000)) +
  theme_minimal() +
  theme(legend.position = "none")
```



First try: classify people with capital gains less than $5000 as low earners and greater than $5000 as high earners.

```
census %>% ggplot(aes(x = capital.gain, y = income)) +
  geom_count(position = position_jitter(width = 0, height = 0.1),
             alpha = 0.5) +
  scale_x_log10(labels = scales::dollar,
                breaks = c(0,1000,5000,100000)) +
  geom_vline(xintercept = 5000, color = "navyblue", lty = 1) +
  theme_minimal() + theme(legend.position = "none")
```



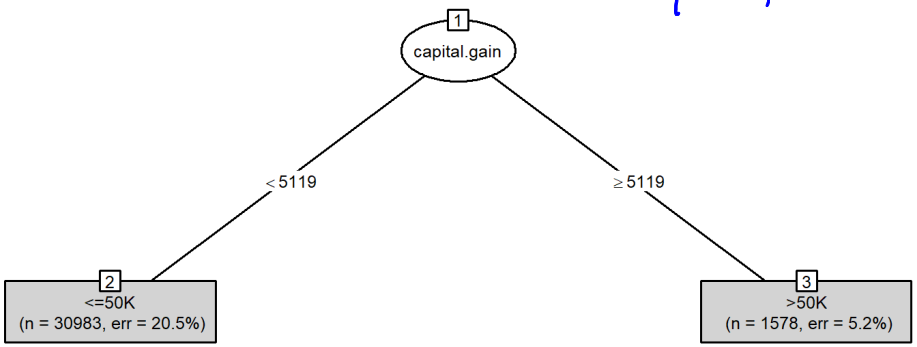
What is the "best" split of the `capital.gain` variable to predict which individuals are high income earners?

Fitting a tree

rpart(y ~ x , data)

```
tree <- rpart(income ~ capital.gain, data = census,
              parms = list(split = "gini"))
plot(as.party(tree),type = "simple", gp = gpar(cex = 0.5))
```

$5,119



R tested all possible thresholds and picked the one with the biggest value for ΔI.

Validation methods

Comparing prediction accuracy

```
predicted.income <- ifelse(census$capital.gain <= 5000,
                           yes="<=50K", no=">50K")
true.income <- census$income
```

Threshold: $5,000

```
##                  true.income
## predicted.income  <=50K  >50K
##           <=50K   24568  6345
##           >50K      152  1496
```

Threshold: $5,119

```
##                  true.income
## predicted.income  <=50K  >50K
##           <=50K   24638  6345
##           >50K       82  1496
```

Which tree is more accurate?

Accuracy: $\frac{24{,}568 + 1496}{\text{total } \#}$

Question

Should we use the same data to build the model and to test it?

↳ No. we want to see how well our tree performs for new data.

Question

Should we use the same data to build the model and to test it?

What can we do instead?

↳ split your data into training and testing sets

↓ build the tree

↓ to test it

Resampling methods in statistics

We've talked about several kinds of randomization

- Randomization test to calculate p-values (two-sample hypothesis tests)
- Bootstrapping to calculate confidence intervals

Resampling methods in statistics

We've talked about several kinds of randomization

- Randomization test to calculate p-values (two-sample hypothesis tests)
- Bootstrapping to calculate confidence intervals

Resampling can also be used to assess the accuracy of a prediction model

↳ divide our sample into two samples
↳ without replacement
↳ training
↳ testing

Validation set approach

Randomly divide data into "training" and "testing" datasets

1. Separate data into training and testing subsets, by randomly selecting rows
    - Ex: 80% training, 20% testing ("hold-out") sample
    - No formal rules for percentages
2. Fit model using training data
3. Run "test" data through the fitted tree and check how many of them are correctly classified

↳ you need to know the values of y to assess prediction accuracy.

Training and Testing Classification Trees

full sample.

```
set.seed(364)
n <- nrow(census) # number of observations in census data
n
```

```
## [1] 32561
```

a random sample with 20% of the numbers between 1 and 32,561

```
# random sample of 20% of row indexes
test_idx <- sample.int(n, size = round(0.2 * n))
# training data is all observations except from training row indexes
train <- census[-test_idx, ]
nrow(train)
```

↳ exclude observations w. index in test_idx.

```
## [1] 26049
```

```
# test data
test <- census[test_idx, ]
nrow(test)
```

↳ include only obs. w index in test_idx

```
## [1] 6512
```

```
train <- train %>% mutate(income = factor(income),
                          workclass = factor(workclass),
                          education = factor(education),
                          marital.status = factor(marital.status),
                          occupation = factor(occupation),
                          relationship = factor(relationship),
                          race = factor(race),
                          sex = factor(sex))

test <- test %>% mutate(income = factor(income),
                          workclass = factor(workclass),
                        education = factor(education),
                          marital.status = factor(marital.status),
                        occupation = factor(occupation),
                          relationship = factor(relationship),
                        race = factor(race),
                          sex = factor(sex))
```

Example: Predicting High Earners

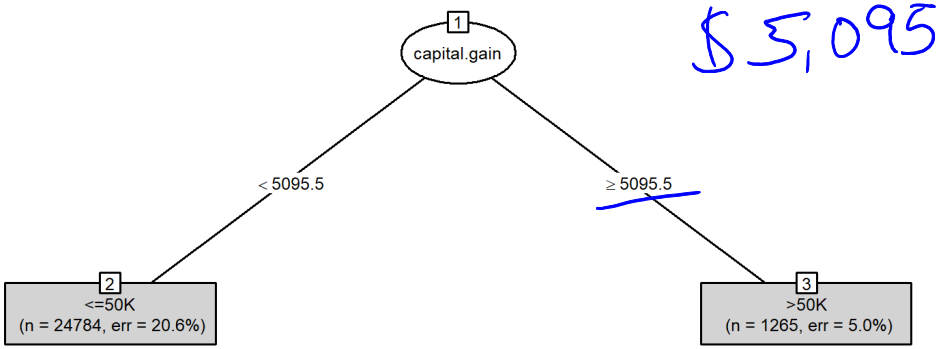
before we used census

```
library(rpart)
tree <- rpart(income ~ capital.gain, data = train,
              parms = list(split = "gini"))
tree
```

```
## n= 26049
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 26049 6317 <=50K (0.75749549 0.24250451)
##   2) capital.gain< 5095.5 24784 5115 <=50K (0.79361685 0.20638315) *
##   3) capital.gain>=5095.5 1265   63 >50K (0.04980237 0.95019763) *
```

Example: Predicting High Earners

```
library(partykit)
plot(as.party(tree),type = "simple", gp = gpar(cex = 0.5))
```



$5,095.5

Is this different from the tree we fit before? Yes, the cutpoint is different.

Why? Used diff. data to build

Is it surprising? No, they are quite close (only 25$)

Example: Predicting High Earners

What is the prediction accuracy of this tree?

use the testing data to check performance

```
predicted_tree <- predict(object = tree, newdata = test,
                          type = "class")
table(predicted_tree,test$income) # confusion matrix
```

```
##
## predicted_tree <=50K >50K
##          <=50K  4969 1230
##          >50K     19  294
```

Which data do we use to measure how "good" our tree is (i.e. prediction accuracy)?

test data

Example: Predicting High Earners

What is the prediction accuracy of this tree?

```
predicted_tree <- predict(object = tree, newdata = test,
                          type = "class")
table(predicted_tree,test$income) # confusion matrix
```

```
##
## predicted_tree <=50K >50K
##          <=50K  4969 1230
##          >50K     19  294
```

$$\frac{4969 + 294}{\text{total}}$$

Which data do we use to measure how "good" our tree is (i.e. prediction accuracy)?

What would happen if you measured the accuracy of a tree using the same data that built it?

↳ overestimate the accuracy.

Limitations of the training/testing set approach

Limitations of the training/testing set approach

- The test's error rate depends on which observations are in the training / validation sets (random)
- Only a subset of observations are used to build the tree
  - Statistical models perform better when more data is used to fit them
  - The validation set approach may **overestimate** the error rate

Accuracy of a classification tree

Consider the confusion matrix:

Predicted	<= 50K	>50K	Total
<=50K	a	b	a+b
>50K	c	d	c+d
Total	a+c	b+d	N = a+b+c+d

Assume we are trying to predict >50K. So this outcome will be considered positive, and <=50K is negative.

- True positive rate (sensitivity): d/(b+d)
- True negative rate (specificity): a/(a+c)
- False positive rate: c/(a+c)
- False negative rate: b/(b+d)
- Accuracy: (a+d)/N

True positive rate (TPR) (sensitivity): Among all observations which are actually "positive", TPR is the proportion for which we predict "positive".

$\Rightarrow$ # actually positive $= b+d$

$\Rightarrow$ sensitivity $= d/b+d$

* just be careful to check which row/column is "positive"/"negative" in each ex.

True negative rate (TNR) (specificity): Among all obs. which are actually "negative", TNR is the proportion for which we predict "negative".

$\Rightarrow$ # actually negative $= a+c$

$\Rightarrow$ specificity $= a/a+c$

Note: a good classifier has high sensitivity and high specificity (higher is better)

False positive rate (FPR) and False negative rate (FNR) are defined similarly.

Note that $TPR + FNR = 1$

and $TNR + FPR = 1$.

↳ look at the formulas & table to understand why this is.

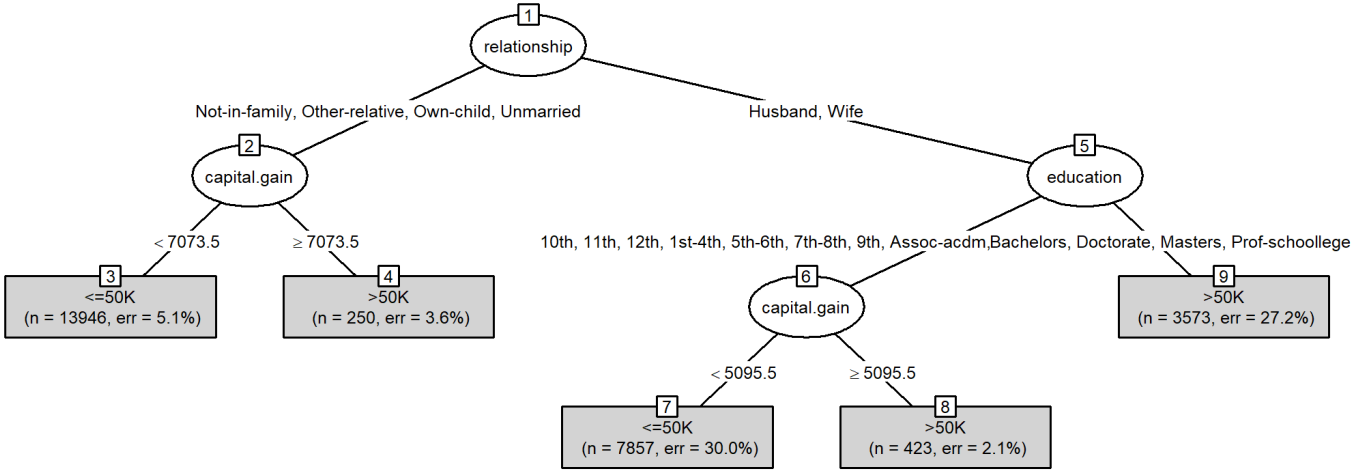
Classification Trees - Adding more variables

We have used only one variable to predict income, but we can use the other variables in the data to try and improve the accuracy.

```
form <- as.formula("income ~ age + workclass + education +
                   marital.status + occupation + relationship +
                   race + sex + capital.gain + capital.loss +
                   hours.per.week")
mod_tree <- rpart(form, data = train)
```

Classification Trees - Adding more variables

```
library(partykit)
plot(as.party(mod_tree),type = "simple", gp = gpar(cex = 0.5))
```



```
mod_tree
```

```
## n= 26049
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 26049 6317 <=50K (0.75749549 0.24250451)
##    2) relationship=Not-in-family,Other-relative,Own-child,Unmarried 14196  947
##      4) capital.gain< 7073.5 13946  706 <=50K (0.94937617 0.05062383) *
##      5) capital.gain>=7073.5 250    9 >50K (0.03600000 0.96400000) *
##    3) relationship=Husband,Wife 11853 5370 <=50K (0.54695014 0.45304986)
##      6) education=10th,11th,12th,1st-4th,5th-6th,7th-8th,9th,Assoc-acdm,Assoc-
##       12) capital.gain< 5095.5 7857 2355 <=50K (0.70026728 0.29973272) *
##       13) capital.gain>=5095.5 423    9 >50K (0.02127660 0.97872340) *
##      7) education=Bachelors,Doctorate,Masters,Prof-school 3573  972 >50K (0.27
```

Accuracy of Classification Tree

```
predicted_mod <- predict(mod_tree, newdata = test, type = "class")
table(predicted_mod, test$income)
```

```
##
## predicted_mod <=50K >50K
##         <=50K  4731  755
##         >50K    257  769
```

- The overall accuracy is: $\frac{4731 + 769}{N} = 0.8446$

$N = 4731 + 755 + 257 + 769 = 6512$

50% threshold

the predict(·) function here is used to get predictions for the test data based on our tree mod_tree (built using training data on p39-40). type=class means we want just the predicted category for each observation in test data, so output is a vector

Accuracy of Classification Tree

Instead of giving just the predicted category, we want to get the predicted probability for each category. Output here is a matrix, w. one row for each observation and one column for each possible category.

Instead of predicting class directly we can predict **probability** of being high income

```
predicted_tree <- predict(object=mod_tree,newdata=test,type="prob")
predicted_tree[c(1,2,6),]
```

```
##       <=50K      >50K
## 1 0.7002673 0.2997327
## 2 0.2720403 0.7279597
## 6 0.0212766 0.9787234
```

can use diff thresholds

↳ predicted probs are related for err rates in each terminal node

Question: where do these predicted probabilities come from?

Look at obs 1 from test data. On p43, we see that based on mod_tree, the probability this individual is low income is 0.70 and the probability they have a high income is 0.30.

↳ From mod_tree (on p40) we can see that this individual must land in terminal node 7. In this node, there are 7857 individuals (from the training data):

↳ 70% of them have low income

↳ 30% of them have high income

(look at the error rate for this terminal node).

Similarly:

- Observation 2 from test set (2nd row on p43) lands in terminal node 9
- Observation 6 from test set (3d row on p43) lands in terminal node 8.

Accuracy of Classification Tree

What if we use 0.5. as threshold

```
# if predicted prob of >50K is >=0.5 then predicted class is >50K
# otherwise predicted class is <=50K
m <- table(predicted_tree[,2] >= 0.5,test$income)
row.names(m) <- c("Pred <50K","Pred >=50K")
m
```

← If probability of high income is ≥0.50, the predicted class is high income, otherwise it is low income.

```
##
##                <=50K >50K
##   Pred <50K     4731  755
##   Pred >=50K     257  769
```

Accuracy: $\frac{4731+769}{N} = 0.8446$

↳ same as on p42, because R uses threshold of 0.50 by default.

Classification Tree Accuracy

If the probability of "high income" is ≥ 0.24, predicted class is "high income", Otherwise predicted class is "low income".

Since only 24% of the sample earns >50K perhaps this is a more sensible cutoff for prediction.

```
predicted_tree <- predict(object = mod_tree, newdata = test,
                          type = "prob")
m <- table(predicted_tree[,2] >= 0.24, test$income)
row.names(m) <- c("Pred <50K", "Pred >=50K")
m
```

```
##
##                <=50K >50K
##    Pred <50K    3370  166
##    Pred >=50K   1618 1358
```

→ different

Accuracy: $\frac{3370 + 1358}{N} = 0.7260.$

↳ lower accuracy than when the threshold was 0.50.

ROC Curves

⇒ Used to compare various classifiers to each other, and help us decide which to use

The ROC curve is a plot of the true positive rate versus the false positive rate for various cut-points.

```
pred <- ROCR::prediction(predictions = predicted_tree[,2],
                         test$income)
perf <- ROCR::performance(pred, 'tpr', 'fpr')
perf_df <- data.frame(perf@x.values, perf@y.values)
names(perf_df) <- c("fpr", "tpr")
roc <- ggplot(data = perf_df, aes(x = fpr, y = tpr)) +
  geom_line(color = "blue") +
  geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf@y.name) + xlab(perf@x.name)
```

output of predict(.., type="prob") ↳ p 45

true values

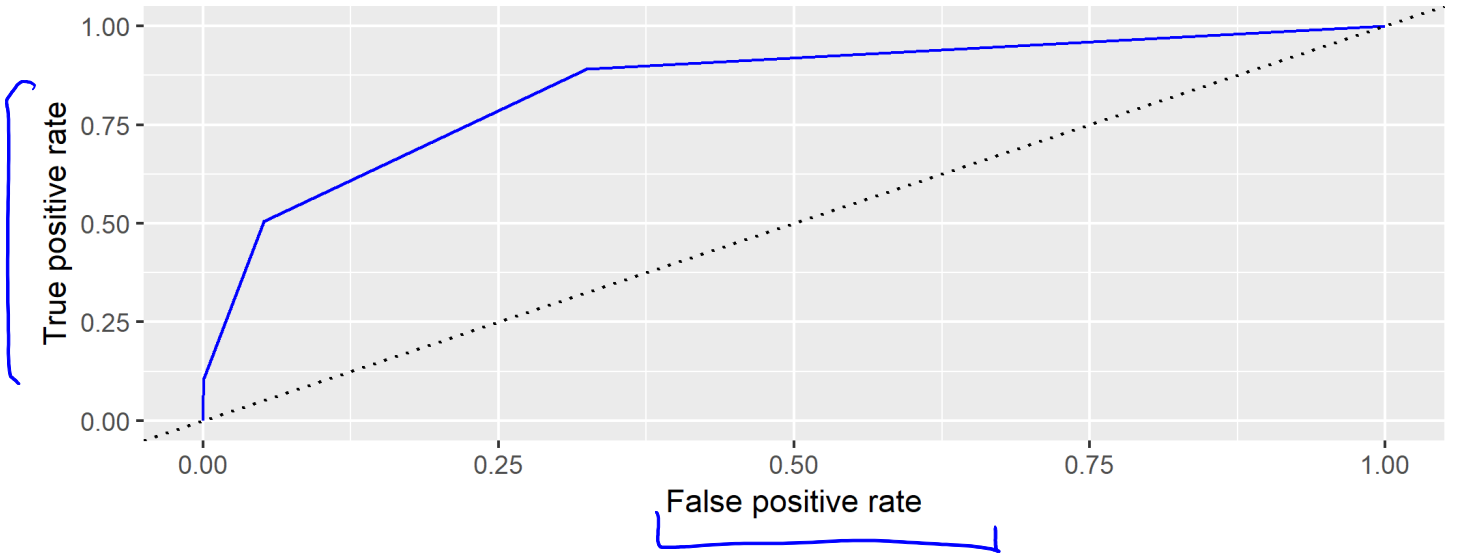
create data frame w TPR and FPR values

plot ROC curve.

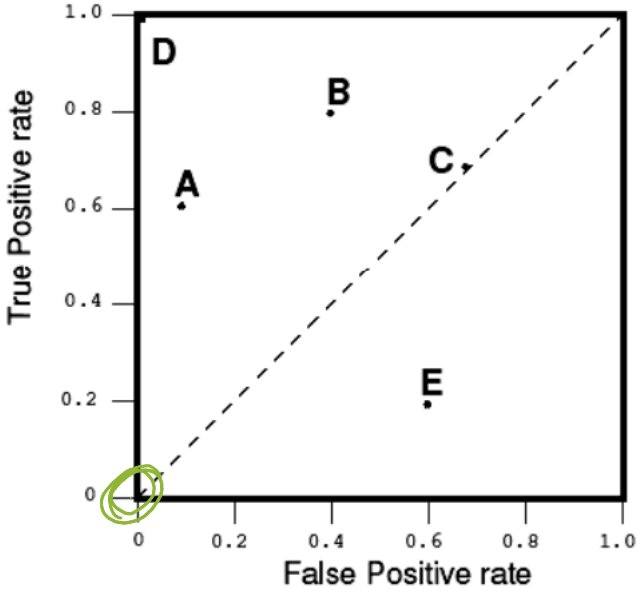
You can copy the code above to make an ROC curve, you just need to change the values in the first two lines (highlighted)

ROC Curves

```
roc
```



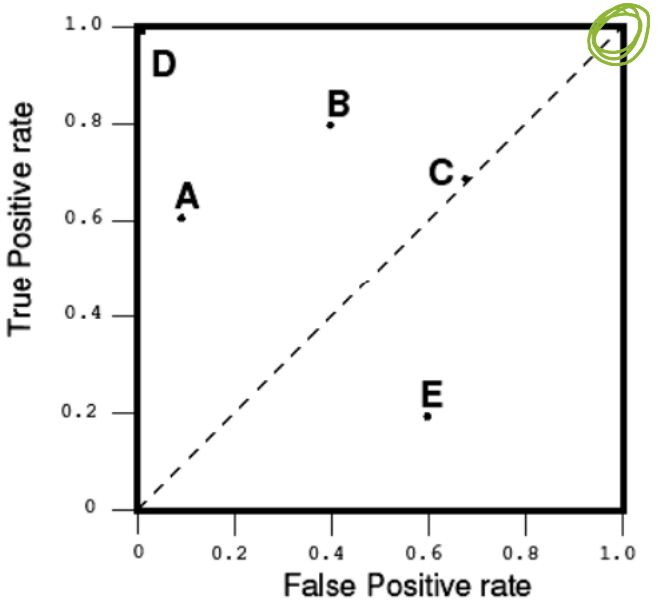
Lower left point



Never classify as 'positive' (i.e. always predict "negative")

- no false positive errors → good
- no true positives → bad
  (because we always predict "negative")

Upper right point

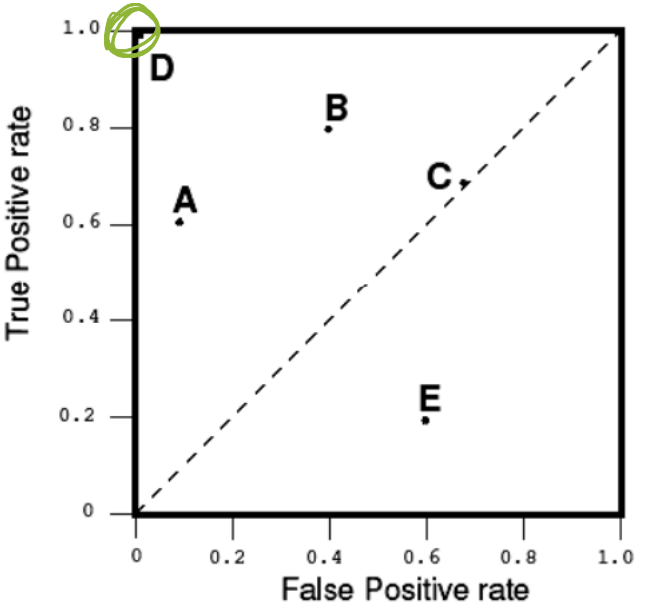


(predict)

Always classify as 'positive'

- all true positives are classified correctly ⇒ good
- none of the true negatives are classified ~~in~~correctly ⇒ bad
  - ↳ i.e all of the obs. which are actually "negative" are predicted incorrectly.

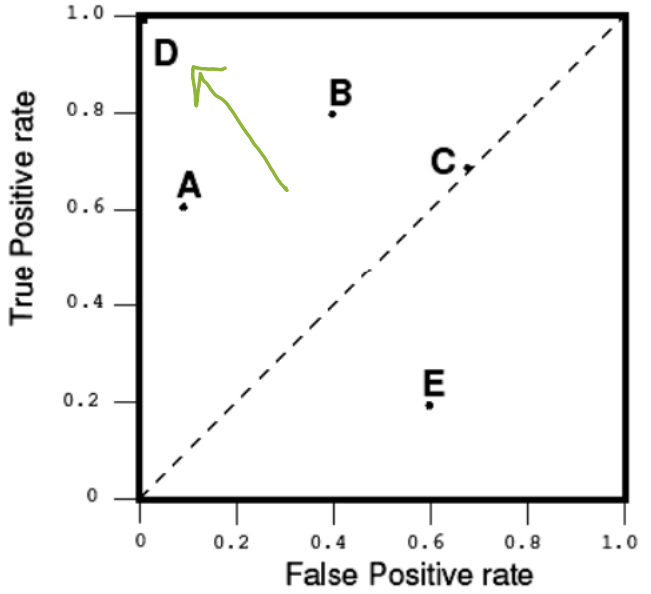
Upper left point (0,1)



Gets everything perfect!

- No false positive errors
- ~~All true positives~~
  No false negative errors

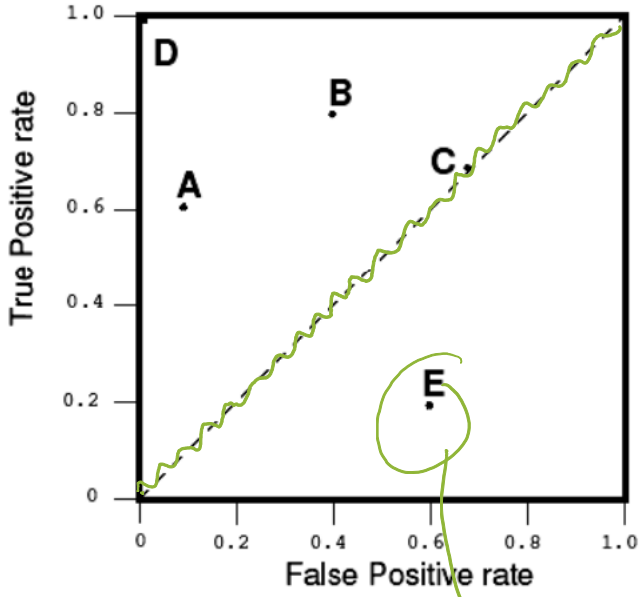
All other points



Better to be close to top left corner

- TP rate is higher
- FP rate is lower
- ideally both!

Point C: random performance



The diagonal line represents the strategy of randomly guessing!

- At (0.7, 0.7), C is guessing the positive class 70\% of the time
  ↳not using information from the predictors.

↳ This classifier's performance is worse than if you just flipped a coin to make the predictions! It is not a useful predictor.

ROC Curves

- The ROC curve is a plot of all possible threshold values for classification.
- The upper-left corner represents a perfect classifier, which would have a true positive rate of 1 and a false positive rate of 0.
- A random classifier would lie along the diagonal, since it would be equally likely to make either kind of mistake.

ROC Curves

The true positive rate and false positive rate for a tree classifier with cutpoint 0.5.

```
predicted_tree <- predict(object = mod_tree, newdata = test,
                          type = "prob")
m <- table(predicted_tree[,2] >= 0.50,test$income)
row.names(m) <- c("Pred <50K","Pred >=50K")
tpr_50 <- m[4]/sum(m[,2])
fpr_50 <- m[2]/sum(m[,1])
tpr_50
```

} calculates TPR and FPR when cutpoint of 0.50 is used

```
## [1] 0.5045932
```

```
fpr_50
```

```
## [1] 0.05152366
```

ROC Curves

The true positive rate and false positive rate for a tree classifier with cutpoint 0.24.

```
predicted_tree <- predict(object = mod_tree, newdata = test,
                          type = "prob")
m <- table(predicted_tree[,2] >= 0.24,test$income)
row.names(m) <- c("Pred <50K","Pred >=50K")
tpr_24 <- m[4]/sum(m[,2])
fpr_24 <- m[2]/sum(m[,1])
tpr_24
```

} calculates TPR and FPR with cutpoint of 0.24

```
## [1] 0.8910761
```

```
fpr_24
```

```
## [1] 0.3243785
```

	cutpoint: 0.50	cutpoint: 0.24
TPR (sensitivity)	0.56	0.89
FPR (1-specificity)	0.05	0.32

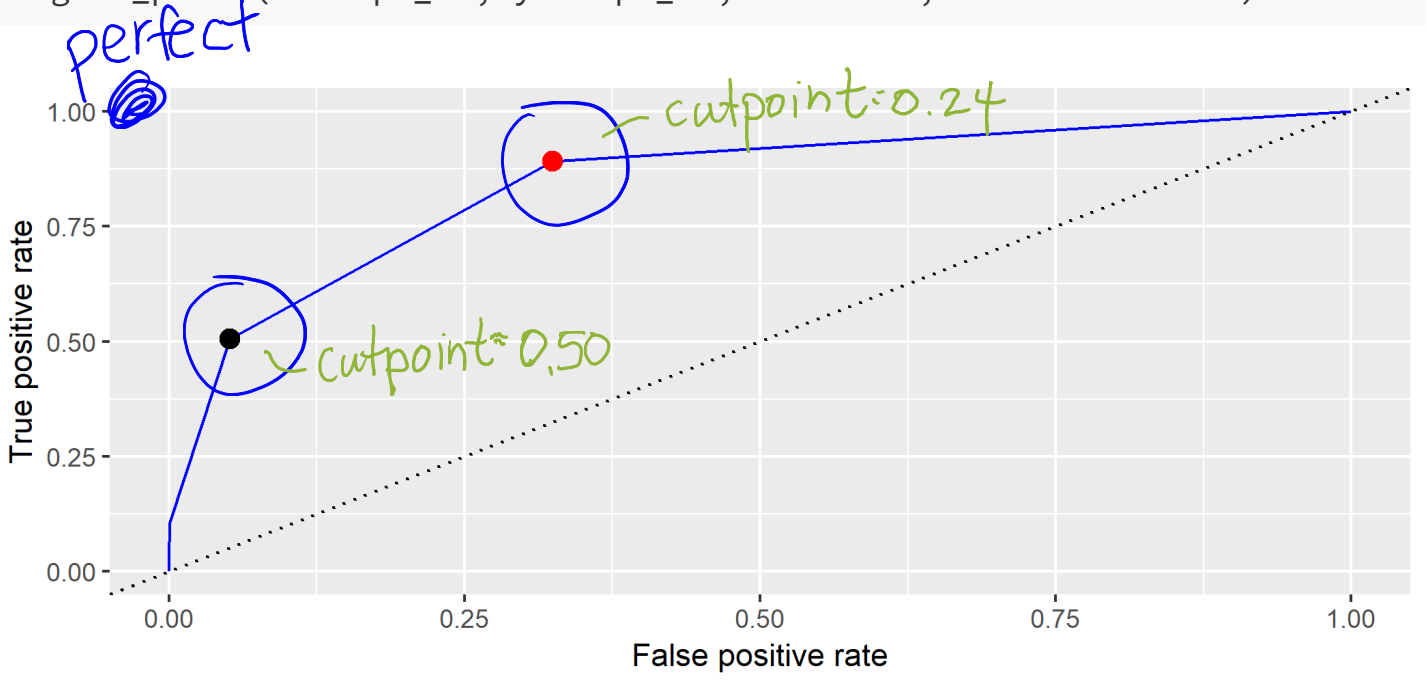
* Cutpoint of 0.24 has higher sensitivity (good!) but also higher FPR, which is equivalent to lower specificity (bad!).

If you care more about making false positive / false negative errors, you may prefer one of the cutpoints over the other.

ROC Curves

The tree with a cutpoint of 0.5 is shown as the black dot and the tree with a cutpoint of 0.24 is shown as the red dot.

```
roc + geom_point(x = fpr_50, y = tpr_50, size = 3, colour = "black") +
  geom_point(x = fpr_24, y = tpr_24, size = 3, colour = "red")
```



Which cutoff is better? they are about the same

Overview

1. Supervised vs unsupervised learning
2. Classification trees
    - Interpretation
    - Methodology
    - Training and testing
    - Accuracy
3. ROC curves